```c
#include <stdio.h>
#include <stdlib.h>

int main() {
   int *arr_malloc, *arr_calloc, *arr_realloc;
   int n = 5, i;

   // --------- malloc() example ---------
   printf("Using malloc() to allocate memory:\n");
   arr_malloc = (int*) malloc(n * sizeof(int));
   if (arr_malloc == NULL) {
      printf("Memory not allocated using malloc().\n");
      exit(1);
   }
   for (i = 0; i < n; i++) {
      arr_malloc[i] = i + 1;
      printf("arr_malloc[%d] = %d\n", i, arr_malloc[i]);
   }

   // --------- calloc() example ---------
   printf("\nUsing calloc() to allocate memory:\n");
   arr_calloc = (int*) calloc(n, sizeof(int));
   if (arr_calloc == NULL) {
      printf("Memory not allocated using calloc().\n");
      exit(1);
   }
   for (i = 0; i < n; i++) {
      printf("arr_calloc[%d] = %d\n", i, arr_calloc[i]);
```

```
    }

    // --------- realloc() example ---------
    printf("\nUsing realloc() to resize memory:\n");
    int new_size = 8;
    arr_realloc = (int*) realloc(arr_malloc, new_size * sizeof(int));
    if (arr_realloc == NULL) {
        printf("Memory not reallocated using realloc().\n");
        free(arr_malloc);
        exit(1);
    }
    // Fill new elements
    for (i = n; i < new_size; i++) {
        arr_realloc[i] = i + 1;
    }
    // Print all elements
    for (i = 0; i < new_size; i++) {
        printf("arr_realloc[%d] = %d\n", i, arr_realloc[i]);
    }

    // --------- Freeing memory ---------
    free(arr_calloc);
    free(arr_realloc);

    printf("\nMemory has been deallocated successfully.\n");

    return 0;
}
```

```
Using malloc() to allocate memory:
arr_malloc[0] = 1
arr_malloc[1] = 2
arr_malloc[2] = 3
arr_malloc[3] = 4
arr_malloc[4] = 5

Using calloc() to allocate memory:
arr_calloc[0] = 0
arr_calloc[1] = 0
arr_calloc[2] = 0
arr_calloc[3] = 0
arr_calloc[4] = 0

Using realloc() to resize memory:
arr_realloc[0] = 1
arr_realloc[1] = 2
arr_realloc[2] = 3
arr_realloc[3] = 4
arr_realloc[4] = 5
arr_realloc[5] = 6
arr_realloc[6] = 7
arr_realloc[7] = 8

Memory has been deallocated successfully.
```